**Exercise 1: Configuring a Basic Spring Application**

**Code:**

**ApplicationContext.xml:**

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd">  
   
 <bean id="bookRepository" class="org.library.repository.BookRepository"/>  
   
 <bean id="bookService" class="org.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository"/>  
 </bean>  
</beans>

**BookRepository:**

public class BookRepository {  
 public void saveBook(String bookName) {  
 System.*out*.println("Book saved: " + bookName);  
 }  
}

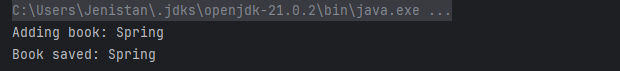
**BookService:**

public class BookService {  
 private BookRepository bookRepository;  
 public void setBookRepository(BookRepository bookRepository)  
 {  
 this.bookRepository = bookRepository;  
 }  
 public void addBook(String bookName) {  
 System.*out*.println("Adding book: " + bookName);  
 bookRepository.saveBook(bookName);  
 }  
}

**MainApp:**

public class App   
{  
 public static void main( String[] args )  
 {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 BookService bookService =context.getBean(BookService.class);  
 bookService.addBook("Spring");  
 }  
}

**Output:**

****

**Exercise 2: Implementing Dependency Injection**

**Code:**

**ApplicationContext.xml:**

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd">  
   
 <bean id="bookRepository" class="org.library.repository.BookRepository"/>  
   
 <bean id="bookService" class="org.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository"/>  
 </bean>  
</beans>

**BookRepository:**

public class BookRepository {

String[] books={"Spring","Golden Hour","Harry Potter"};

public void getAllBooks(){  
 for (String s:books){  
 System.*out*.println(s);  
 }  
}  
}

**BookService:**

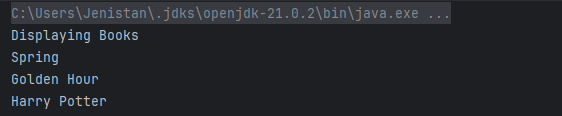
public class BookService {  
 private BookRepository bookRepository;  
 public void setBookRepository(BookRepository bookRepository)  
 {  
 this.bookRepository = bookRepository;  
 }  
 public void displayBooks() {  
 System.*out*.println("Displaying Books");  
 bookRepository.getAllBooks(); // Use the repository  
}

}

**MainApp:**

public class App   
{  
 public static void main( String[] args )  
 {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 BookService bookService =context.getBean(BookService.class);  
 bookService.displayBooks();   
 }  
}

**Output:**

****

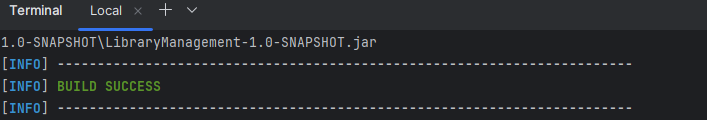
**Exercise 4: Creating and Configuring a Maven Project**

**Code:**

**ApplicationContext.xml:**

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>org.library</groupId>  
 <artifactId>LibraryManagement</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>LibraryManagement</name>  
 <url>http://maven.apache.org</url>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <maven.compiler.source>1.8</maven.compiler.source>  
 <maven.compiler.target>1.8</maven.compiler.target>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>3.8.1</version>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>7.0.0-M6</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>5.3.32</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>5.3.32</version>  
 </dependency>  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-api</artifactId>  
 <version>1.7.30</version>  
 </dependency>  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-simple</artifactId>  
 <version>1.7.30</version>  
 </dependency>  
 </dependencies>  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.8.1</version>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

**Output:**

****

**Additional HandsOn:**

**Exercise 5: Configuring the Spring IoC Container**

**Code:**

**ApplicationContext.xml:**

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd">  
   
 <bean id="bookRepository" class="org.library.repository.BookRepository"/>  
   
 <bean id="bookService" class="org.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository"/>  
 </bean>  
</beans>

**BookRepository:**

public class BookRepository {

String[] books={"Spring","Golden Hour","Harry Potter"};

public void getAllBooks(){

System.*out*.println("Getting All books from the Repository...");  
 for (String s:books){  
 System.*out*.println(s);  
 }  
}  
}

**BookService:**

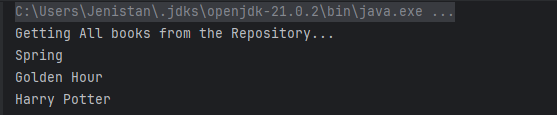
public class BookService {  
 private BookRepository bookRepository;  
 public void setBookRepository(BookRepository bookRepository)  
 {  
 this.bookRepository = bookRepository;  
 }  
 public void showBooks() {  
 bookRepository.getAllBooks(); // Use the repository  
}

}

**MainApp:**

public class App   
{  
 public static void main( String[] args )  
 {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 BookService bookService =context.getBean(BookService.class);  
 bookService.showBooks();   
 }  
}

**Output:**

****

**Exercise 7: Implementing Constructor and Setter Injection**

**Code:**

**ApplicationContext.xml:**

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd">  
   
 <bean id="bookRepository" class="org.library.repository.BookRepository"/>  
   
 <bean id="bookService" class="org.library.service.BookService">

<constructor-arg value="Central City Library" />  
 <property name="bookRepository" ref="bookRepository"/>  
 </bean>  
</beans>

**BookRepository:**

public class BookRepository {

String[] books={"Spring","Golden Hour","Harry Potter"};

public void getAllBooks(){

System.*out*.println("Getting All books from the Repository...");  
 for (String s:books){  
 System.*out*.println(s);  
 }  
}  
}

**BookService:**

public class BookService {  
 private BookRepository bookRepository;

private String libraryName;  
  
// Constructor for constructor injection  
public BookService(String libraryName) {  
 this.libraryName = libraryName;  
}

public void setBookRepository(BookRepository bookRepository)  
 {  
 this.bookRepository = bookRepository;  
 }  
 public void showBooks() {

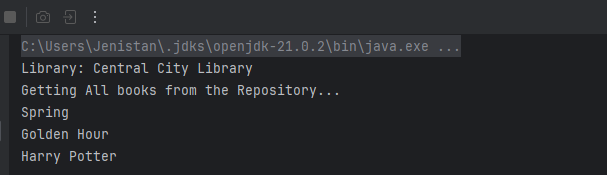
System.*out*.println("Library: " + libraryName);  
 bookRepository.getAllBooks(); // Use the repository  
}

}

**MainApp:**

public class App   
{  
 public static void main( String[] args )  
 {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 BookService bookService =context.getBean(BookService.class);  
 bookService.showBooks();   
 }  
}

**Output:**

****

**Exercise 9: Creating a Spring Boot Application**

**Code:**

**Book.java:**

package com.books.Library;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
  
@Entity  
public class Book {  
 @Id  
 private Long id;  
 private String title;  
 private String author;  
 public Long getId() { return id; }// Getters and setters  
 public void setId(Long id) { this.id = id; }  
 public String getTitle() { return title; }  
 public void setTitle(String title) { this.title = title; }  
 public String getAuthor() { return author; }  
 public void setAuthor(String author) { this.author = author; }  
}

**BookRepository.java:**

package com.books.Library;  
  
import com.books.Library.Book;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface BookRepository extends JpaRepository<Book, Long> {  
}

**BookController.java:**

package com.books.Library;  
  
import com.books.Library.Book;  
import com.books.Library.BookRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/books")  
public class BookController {  
 @Autowired  
 private BookRepository bookRepository;  
 @GetMapping  
 public List<Book> getAllBooks() {  
 return bookRepository.findAll();  
 }  
 @PostMapping  
 public Book addBook(@RequestBody Book book) {  
 return bookRepository.save(book);  
 }  
 @PutMapping("/{id}")  
 public Book updateBook(@PathVariable Long id, @RequestBody Book book) {  
 book.setId(id);  
 return bookRepository.save(book);  
 }  
 @DeleteMapping("/{id}")  
 public void deleteBook(@PathVariable Long id) {  
 bookRepository.deleteById(id);  
 }  
}

**Application.properties:**

spring.application.name=Library  
spring.datasource.url=jdbc:h2:mem:librarydb  
spring.datasource.driverClassName=org.h2.Driver  
spring.datasource.username=sa  
spring.datasource.password=  
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect  
spring.h2.console.enabled=true  
spring.jpa.hibernate.ddl-auto=update

**Output:**

**Add a Book – POST /books**

**URL**: http://localhost:8080/books  
**Method**: POST  
**Body** → JSON (raw):

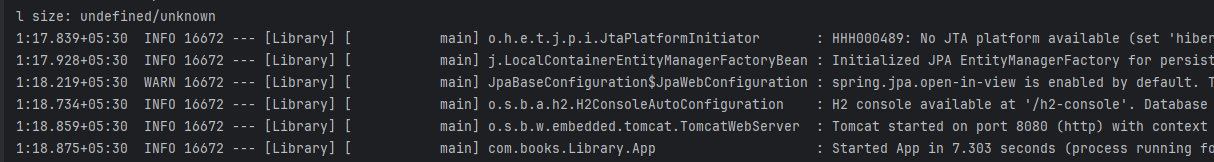
{

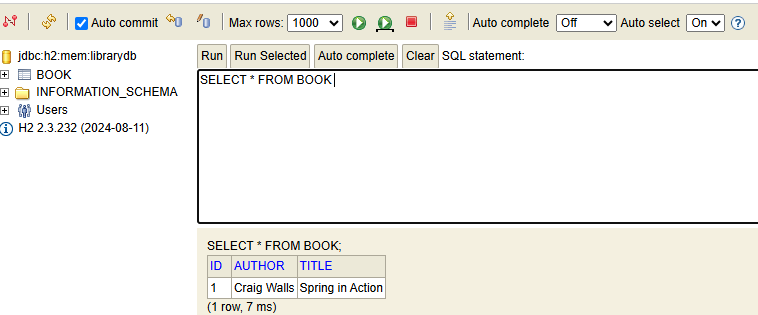
"id": 1,

"title": "Spring in Action",

"author": "Craig Walls"

}

****

****